LiX Beamline Manual

(last revised 2016 Mar 20)

1. Controls

1.1) Introduction

Lix has four PC's in the control room, three of them have linux system and one windows. Data acquisition and hardware motion is primarily controlled by Bluesky framework, it is driven by python objects and is basically a command line interface. Additionally GUI screens are available created through control system studio (css) to move the motors and or do simple counts.

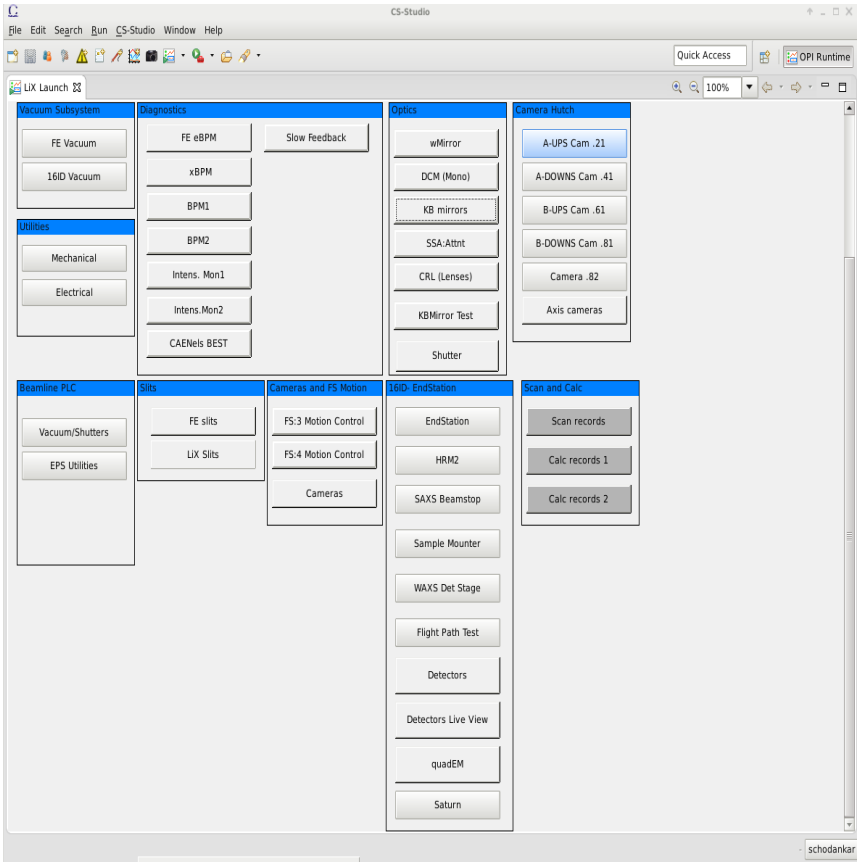
1.2) Bluesky startup

To start a bluesky data acquisition session open a terminal window and login to xf16idc-gpu1. Activate the session by **"bsui"**

In Ipython command line type **login()**, followed by username, proposal number and folder name. This will create necessary folder and sets path for saving the data. Data is usually saved in /GPFS/xf16id/exp_path/proposal number/folder name/files

1.3) CSS screens

User interface for data collection and motion control is available through control system studio. To launch css go to any terminal and execute "**run-css**" this will launch LIX css main page



It contains links to various interfaces such as detector screens, cameras and other controls.

1.4) Basic Bluesky commands

All units are in mm or degrees by default.

A. High-throughput Solution scattering

Move the PCR tube holder:	
sol.select_tube_pos('park')	move the holder out for retrieval
Measure one sample	
sol.measure(tn, vol=XX, exp=XX, repeats=XX, sample_name='test')	tn: tube number (1-18) nd:needle, either 'upstream' or 'downstream' vol: volume to measure, the actual loaded volume is vol + sol.sample_headroom, which is 13 uL by default but can be revised

	exp:exposure time, 5 sec is typical repeats: number of scattering patterns to take from the sample sample_name:if the sample name is not given, 'test' is assumed
Other commands (not normally used by users)	
sol.select_flow_cell(cn)	Cn should be one of 'top', 'middle' or 'bottom'. This is done automatically when running sol.measure(), as described below
sol.move_tube_holder(pos)	pos should be either 'up' or 'down', this is only allowed when the tube holder is aligned with the sample needles
sol.select_tube_pos(pos)	Move the holder to tube #pos, pos=1-18, 1 is on the inboard side; pos=0 is the drain
sol.wash_needle(nd,repeats=3, dry_duration=55)	Washes needle (nd= upstream or downstream)
sol.prepare_to_load_sample(tn)	Prepares to load sample at position tn(1-18)
sol.load_sample(vol)	Loads sample in the position decided by prepare to load sample command
sol.collect_data(vol, exp, repeats, sample_name='test')	A given volume of sample is flown through the flow cell while x-ray data is collected for n exposure secs and repeated for a given amount.
sol.return_sample()	Sample is return back to pcr tubes

Interrupting Scans

A scan can be interrupted by pressing Ctrl+c twice.

What to do after interrupting
RE.resume() - to resume the scan back
RE.stop() - to state the scan is successful and come out of the measurement
RE.abort() - to abort the scan

Executing a batch file

A batch file can be runned to measure a sequence of samples using command
"%run -i /GPFS/Commissioning/folder_path/filename"

A typical example for a batch file is
sol.measure(2,vol=45,exp=5,repeats=3,sample_name="sample1")
sol.measure(4,vol=45,exp=5,repeats=3,sample_name="sample2")
sol.select_tube_pos("park")

B. In-line size exclusion chromatography

Software controlling HPLC setup is installed on a separate windows computer. Therefore HPLC x-ray data collection is two part process, executing the hplc parameters on the windows computer and starting the sample elution.
On linux machine data collection is synchronized by executing command
"collect_hplc("sample_name",exp)"
This will send the signal to the HPLC machine that beamline is ready for data acquisition, on receiving the command HPLC software starts the pretreatment and running the sample as soon as the sample starts to flow, a second signal is send for starting the data collection at beamline side.

The pressure difference between the column to the x-ray flow cell can cause the soluble gas in the buffer to release giving rise to bubbles in the flow cell. HPLC flow cell is equipped with a bubble remover tool. Currently the process is not automatized, so the user has to keep an eye on the image as soon as the bubble appears it can be sucked out using the following command
"sol.ctrl.sv_bubble.put(n)"
n of 1 or 2 is enough to remove the bubble a higher number can be problematic as it would suck up the sample too.

Triggering data collection manually
Use the following command
caput(XF:16IDC-ES:Sol{ctrl}HPLC_bypass, 1)

Data processing

Users can download the pyXS library needed for re-processing their data from the beamline Google Drive: https://drive.google.com/drive/folders/0B55zDKEfszcwdThzU1d0NFhfOHc .

pyXS is a collection of python scripts for processing x-ray scattering data. Compared to its original version from NSLS beamline X9, it now can deal with data from more than two detectors that collect data simultaneously. For compatibility with the Pilatus detectors at LiX, pyXS is now using FabIO to read image files. We are also migrating to python3, with the rest of NSLS-II.

In order for pyXS to run on your computer, you will need the essential python modules such as PIL/pillow, numpy and matplotlib. The easiest way to get all necessary python modules in one shot is to get Anaconda (https://www.continuum.io/downloads), which includes most modules already. If you are worried about disk space, install Miniconda (http://conda.pydata.org/miniconda.html) instead, in which case you will need to install additional modules after conda is installed:

```
conda install numpy scipy matplotlib pillow ipython-notebook
```

FabIO is not initially installed under Anaconda. You can install it using pip:

```
pip install fabio
```

Once you unpack pyXS, run the following command within the pyXS directory:

```
python setup.py install
```

This will install pyxs into the site-packages directory so that python can find it when you import pyxs from your script. The real space-reciprocal space conversion within pyXS is performed by code written in C. The RQconv module may need to be compiled, if the binary has not been built (the distribution contains binaries for Linux and Windows, 64-bit python 3.5). The setup script will compile the RQconv module as well. But for that to work you will need swig (can be installed by anaconda) and an appropriate compiler. This should not be an issue for Linux and Mac. For Windows the C/C++ compiler can be downloaded from http://landinghub.visualstudio.com/visual-cpp-build-tools .

The distribution (under pyXS/Example) contains example data and an ipython notebook for processing them. To start ipython notebook, simply run “ipython notebook” or “jupyter notebook” from a terminal, or command prompt for Windows. This will open a browser showing files in the local directory. Therefore it may be a good idea to start ipython from a directory under which the notebook is located.